Satisfiability

- **Input**: Set of clauses (Convert KB to conjunctive normal form (CNF))
- **Output**: Truth assignment that satisfies all clauses, or failure
- The paradigmatic NP-complete problem
- **Solution**: Search
  - Two main approaches:
    - **Backtracking** (e.g.: DPLL)
    - **Stochastic local search** (e.g.: WalkSAT)

**Backtracking**

- Assign truth values by depth-first search
- Assigning a variable deletes false literals and satisfied clauses
- Empty set of clauses: Success
- Empty clause: Failure
- Additional improvements:
  - **Unit propagation** (unit clause forces truth value)
  - **Pure literals** (same truth value everywhere)

**The DPLL Algorithm**

```plaintext
if CNF is empty then
    return true
else if CNF contains an empty clause then
    return false
else if CNF contains a pure literal x then
    return DPLL(CNF(x))
else if CNF contains a unit clause (u) then
    return DPLL(CNF(u))
else
    choose a variable x that appears in CNF
    if DPLL(CNF(x)) = true then return true
    else return DPLL(CNF(not x))
```

**Stochastic Local Search**
- Uses complete assignments instead of partial
- Start with random state
- Flip variables in unsatisfied clauses
- Hill-climbing: Minimize # unsatisfied clauses
- Avoid local minima: Random flips
- Multiple restarts

**The WalkSAT Algorithm**

```plaintext
for i ← 1 to max-tries do
    solution = random truth assignment
    for j ← 1 to max-flips do
        if all clauses satisfied then
            return solution
        end if
        c ← random unsatisfied clause
        with probability p
            flip a random variable in c
        else
            flip variable in c that maximizes number of satisfied clauses
        end if
    end for
return failure
```

2